

The Continuing Arms Race

ACM Books

Editor in Chief

M. Tamer Özsu, *University of Waterloo*

ACM Books is a new series of high-quality books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers. ACM Books publications are widely distributed in both print and digital formats through booksellers and to libraries (and library consortia) and individual ACM members via the ACM Digital Library platform.

The Continuing Arms Race: Code-Reuse Attacks and Defenses

Editors: Per Larsen, *Immunant, Inc.*

Ahmad-Reza Sadeghi, *Technische Universität Darmstadt*

2018

Frontiers of Multimedia Research

Editor: Shih-Fu Chang, *Columbia University*

2018

Shared-Memory Parallelism Can Be Simple, Fast, and Scalable

Julian Shun, *University of California, Berkeley*

2017

Computational Prediction of Protein Complexes from Protein Interaction Networks

Sriganesh Srihari, *The University of Queensland Institute for Molecular Bioscience*

Chern Han Yong, *Duke-National University of Singapore Medical School*

Limsoon Wong, *National University of Singapore*

2017

The Handbook of Multimodal-Multisensor Interfaces, Volume 1: Foundations, User Modeling, and Common Modality Combinations

Editors: Sharon Oviatt, *Incaa Designs*

Björn Schuller, *University of Passau and Imperial College London*

Philip R. Cohen, *Voicebox Technologies*

Daniel Sonntag, *German Research Center for Artificial Intelligence (DFKI)*

Gerasimos Potamianos, *University of Thessaly*

Antonio Krüger, *German Research Center for Artificial Intelligence (DFKI)*

2017

Communities of Computing: Computer Science and Society in the ACM

Thomas J. Misa, Editor, *University of Minnesota*

2017

Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining

ChengXiang Zhai, *University of Illinois at Urbana-Champaign*

Sean Massung, *University of Illinois at Urbana-Champaign*

2016

An Architecture for Fast and General Data Processing on Large Clusters

Matei Zaharia, *Stanford University*

2016

Reactive Internet Programming: State Chart XML in Action

Franck Barbier, *University of Pau, France*

2016

Verified Functional Programming in Agda

Aaron Stump, *The University of Iowa*

2016

The VR Book: Human-Centered Design for Virtual Reality

Jason Jerald, *NextGen Interactions*

2016

Ada's Legacy: Cultures of Computing from the Victorian to the Digital Age

Robin Hammerman, *Stevens Institute of Technology*

Andrew L. Russell, *Stevens Institute of Technology*

2016

Edmund Berkeley and the Social Responsibility of Computer Professionals

Bernadette Longo, *New Jersey Institute of Technology*

2015

Candidate Multilinear Maps

Sanjam Garg, *University of California, Berkeley*

2015

Smarter Than Their Machines: Oral Histories of Pioneers in Interactive Computing

John Cullinane, *Northeastern University; Mossavar-Rahmani Center for Business and Government, John F. Kennedy School of Government, Harvard University*

2015

A Framework for Scientific Discovery through Video Games

Seth Cooper, *University of Washington*

2014

Trust Extension as a Mechanism for Secure Code Execution on Commodity Computers

Bryan Jeffrey Parno, *Microsoft Research*

2014

Embracing Interference in Wireless Systems

Shyamnath Gollakota, *University of Washington*

2014

The Continuing Arms Race

Code-Reuse Attacks and Defenses

Per Larsen

Immunant, Inc.

Ahmad-Reza Sadeghi

Technische Universität Darmstadt

ACM Books #18



Copyright © 2018 by the Association for Computing Machinery
and Morgan & Claypool Publishers

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopy, recording, or any other except for brief quotations in printed reviews—without the prior permission of the publisher.

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances in which Morgan & Claypool is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

The Continuing Arms Race

Per Larsen, Ahmad-Reza Sadeghi, editors

books.acm.org

www.morganclaypoolpublishers.com

ISBN: 978-1-97000-183-9 hardcover

ISBN: 978-1-97000-180-8 paperback

ISBN: 978-1-97000-181-5 eBook

ISBN: 978-1-97000-182-2 ePub

Series ISSN: 2374-6769 print 2374-6777 electronic

DOIs:

10.1145/3129743 Book	10.1145/3129743.3129749 Chapter 5
10.1145/3129743.3129744 Preface	10.1145/3129743.3129750 Chapter 6
10.1145/3129743.3129745 Chapter 1	10.1145/3129743.3129751 Chapter 7
10.1145/3129743.3129746 Chapter 2	10.1145/3129743.3129752 Chapter 8
10.1145/3129743.3129747 Chapter 3	10.1145/3129743.3129753 References
10.1145/3129743.3129748 Chapter 4	

A publication in the ACM Books series, #18

Editor in Chief: M. Tamer Özsu, *University of Waterloo*

First Edition

10 9 8 7 6 5 4 3 2 1

Contents

Preface xi

Chapter 1 How Memory Safety Violations Enable Exploitation of Programs 1

Mathias Payer

- 1.1 Memory Safety 4
- 1.2 Data Integrity 8
- 1.3 Confidentiality 10
- 1.4 Data-Flow and Control-Flow Integrity 11
- 1.5 Policy Enforcement 15
- 1.6 An Adversary's Toolkit 16
- 1.7 Conclusion 22

Chapter 2 Protecting Dynamic Code 25

Gang Tan, Ben Niu

- 2.1 Overview of Challenges and Solutions 26
- 2.2 Type-Based CFG Generation 28
- 2.3 Handling Dynamically Linked Libraries 39
- 2.4 Handling Just-In-Time Compiled Code 48
- 2.5 Related Work 58
- 2.6 Conclusion 60

Chapter 3 Diversity and Information Leaks 61

Stephen Crane, Andrei Homescu, Per Larsen, Hamed Okhravi, Michael Franz

- 3.1 Software Diversity 62
- 3.2 Information Leakage 63

- 3.3 Mitigating Information Leakage 64
- 3.4 Address Oblivious Code Reuse 69
- 3.5 Countering Address-Oblivious Code Reuse 70
- 3.6 Evaluation of Code-Pointer Authentication 74
- 3.7 Conclusion 78

Chapter 4 Code-Pointer Integrity 81

*Volodymyr Kuznetsov, László Szekeres, Mathias Payer,
George Candea, R. Sekar, Dawn Song*

- 4.1 Introduction 81
- 4.2 Related Work 84
- 4.3 Threat Model 87
- 4.4 Design 87
- 4.5 The Formal Model of CPI 97
- 4.6 Implementation 102
- 4.7 Evaluation 109
- 4.8 Conclusion 116

Chapter 5 Evaluating Control-Flow Restricting Defenses 117

*Enes Göktas, Elias Athanasopoulos, Herbert Bos,
Georgios Portokalidis*

- 5.1 Introduction 117
- 5.2 Control-Flow Restricting Defenses 119
- 5.3 Security Analysis 122
- 5.4 Quantifying Gadget Availability in CFR 127
- 5.5 Proof-of-Concept Exploit against CFR 131
- 5.6 Summary 135

Chapter 6 Attacking Dynamic Code 139

Felix Schuster, Thorsten Holz

- 6.1 Goals and Attacker Model 140
- 6.2 Counterfeit Object-Oriented Programming 142
- 6.3 Loopless Counterfeit Object-Oriented Programming 157
- 6.4 A Framework for Counterfeit Object-Oriented Programming 160
- 6.5 Proof-of-Concept Exploits 162
- 6.6 Discussion 168
- 6.7 Security Assessment of Existing Defenses 173
- 6.8 Conclusion 179

Chapter 7 Hardware Control Flow Integrity 181

*Yier Jin, Dean Sullivan, Orlando Arias, Ahmad-Reza Sadeghi,
Lucas Davi*

- 7.1 Introduction 181
- 7.2 Threat Model and Assumptions 184
- 7.3 Requirements 185
- 7.4 Modeling CFI 186
- 7.5 Constructing a Precise Stateful CFI Policy 190
- 7.6 Hardware-Enhanced CFI: Design and Implementation 192
- 7.7 Security Evaluation 200
- 7.8 Performance Evaluation 205
- 7.9 Related Work 208
- 7.10 Conclusion 210

Chapter 8 Multi-Variant Execution Environments 211

Bart Coppens, Bjorn De Sutter, Stijn Volckaert

- 8.1 General Design of an MVEE 212
- 8.2 Implementation of GHUMVEE 217
- 8.3 Inconsistencies and False Positive Detections 220
- 8.4 Comprehensive Protection against Code-Reuse Attacks 233
- 8.5 Relaxed Monitoring 241
- 8.6 Evaluation 250
- 8.7 Conclusion 259

References 261

Contributor Biographies 283

Preface

Our societies are becoming increasingly dependent on emerging technologies and connected computer systems that are increasingly trusted to store, process, and transmit sensitive data. While generally beneficial, this shift also raises many security and privacy challenges. The growing complexity and connectivity offers adversaries a large attack surface. In particular, the connection to the Internet facilitates remote attacks without the need for physical access to the targeted computing platforms. Attackers exploit security vulnerabilities in modern software with the ultimate goal of taking control over the underlying computing platforms. There are various causes of these vulnerabilities, the foremost being that the majority of software (including operating systems) is written in unsafe programming languages (mainly C and C++) and by developers who are by-and-large not security experts.

Memory errors are a prominent vulnerability class in modern software: they persist for decades and still are used as the entry point for today's state-of-the-art attacks. The canonical example of a memory error is the stack-based buffer overflow vulnerability, where the adversary overflows a local buffer on the stack, and overwrites a function's return address. While modern defenses protect against this attack strategy, many other avenues for exploitation exist, including those that leverage heap, format string, or integer overflow vulnerabilities.

Given a memory vulnerability in the program, the adversary typically provides a malicious input that exploits this vulnerability to trigger malicious program actions not intended by the benign program. This class of exploits aims to hijack the control flow of the program and differs from conventional malware, which encapsulates the malicious code inside a dedicated executable that needs to be executed on the target system and typically requires no exploitation of a program bug.

As mentioned above, the continued success of these attacks is mainly attributed to the fact that large portions of software programs are implemented in type-unsafe languages (C, C++, or Objective-C) that do not guard against malicious program inputs using bounds checking, automatic memory management, etc. However, even

type-safe languages like Java rely on virtual machines and complex runtimes that are in turn implemented in type-unsafe languages out of performance concerns. Unfortunately, as modern applications grow more complex, memory errors and vulnerabilities will likely continue to exist, with no end in sight.

Regardless of the attacker's method of choice, exploiting a vulnerability and gaining control over an application's control flow is only the first step of an attack. The second step is to change the behavior of the compromised application to perform malicious actions. Traditionally, this has been realized by injecting malicious code into the application's address space, and later executing the injected code. However, with the widespread enforcement of data execution prevention (DEP), such attacks are more difficult to launch today. Unfortunately, the long-held assumption that only code injection posed a risk was shattered with the introduction of code-reuse attacks, such as return-into-libc and return-oriented programming (ROP). As the name implies, code-reuse attacks do not require any code injection and instead repurpose benign code already resident in memory.

Code-reuse techniques are applicable to a wide range of computing platforms: x86-based platforms, embedded systems running on an Atmel AVR processor, mobile devices based on ARM, PowerPC-based Cisco routers, and voting machines deploying a z80 processor. Moreover, the powerful ROP technique is Turing-complete, i.e., it allows an attacker to execute arbitrary malicious code.

In fact, the majority of state-of-the-art run-time exploits leverage code-reuse attack techniques, e.g., against Internet Explorer, Apple QuickTime, Adobe Reader, Microsoft Word, or the GnuTLS library. Even large-scale cyberattacks such as the popular Stuxnet worm, which damaged Iranian centrifuge rotors, incorporated code-reuse attack techniques.

Indeed, even after more than three decades, memory corruption exploits remain a clear and present danger to the security of modern software and hardware platforms. This is why the research community both in academia and industry have invested major efforts in the recent past to mitigate the threat. Various defenses have been proposed and even deployed by Google, Microsoft, Intel, etc. The most prominent defenses are based on enforcement (e.g., Control-Flow Integrity [CFI]) or randomization (also known as software diversity) of certain program aspects. Both types of defenses have distinct advantages and disadvantages. Randomization makes it hard for attackers to chain together their attack gadgets, is efficient, and can be applied to complex software such as web browsers. It can have different levels of granularity, from a simple Address Space Layout Randomization (ASLR) to fine-grained randomization at function or even instruction level. However, randomization requires high entropy and all randomization schemes are inherently

vulnerable to information leakage. CFI, on the other hand, provides guarantees that the application does not deviate from the intended program flow, yet it requires a security and efficiency tradeoff: i.e., coarse-grained CFI have been shown to be vulnerable, and fine-grained CFI can be inefficient without hardware support. Researchers have been working on improving these schemes with novel ideas in both software and hardware design. Many proposed defenses have been bypassed by other attacks, generating a large body of literature on this topic.

It seems that the arms race between attackers and defenders continues. Although researchers have raised the bar for adversaries, there are still a number of challenges to tackle against sophisticated attacks. Despite all the recent proposals on various defenses, we cannot claim that the problem is entirely solved. However, our community has gained much insight through recent results on how and to what extent we need to employ certain design principles to significantly reduce the effect of code-reuse attacks.

The main purpose of this book is to provide readers with some of the most influential works on run-time exploits and defenses. We hope that this material will inspire readers and generate new ideas and paradigms.

Per Larsen

Ahmad-Reza Sadeghi

February 2018

References

- M. Abadi, M. Budiu, Ú. Erlingsson, and J. Ligatti. 2005a. Control-flow integrity: Principles, implementations, and applications. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, pp. 340–353. DOI: [10.1145/1102120.1102165](https://doi.org/10.1145/1102120.1102165). [12](#), [25](#), [38](#), [39](#), [62](#), [82](#), [86](#), [95](#), [97](#), [110](#), [114](#), [117](#), [139](#), [141](#), [173](#), [174](#), [181](#), [186](#), [211](#), [233](#), [243](#), [249](#)
- M. Abadi, M. Budiu, Ú. Erlingsson, and J. Ligatti. 2005b. A theory of secure control flow. In *Proceedings of the 7th International Conference on Formal Methods and Software Engineering (ICFEM)*. DOI: [10.1007/11576280_9](https://doi.org/10.1007/11576280_9). [182](#), [186](#)
- M. Abadi, M. Budiu, Ú. Erlingsson, and J. Ligatti. 2009. Control-flow integrity: Principles, implementations, and applications. *ACM Trans. Inf. Syst. Secur.*, 13(1). DOI: [10.1145/1609956.1609960](https://doi.org/10.1145/1609956.1609960). [181](#), [189](#), [208](#)
- A. Acharya and M. Raje. 2000. MAPbox: Using parameterized behavior classes to confine untrusted applications. In *Proceedings of the 9th USENIX Security Symposium (SSYM)*, pp. 1–17. [16](#)
- P. Akrividis. 2010. Cling: A memory allocator to mitigate dangling pointers. In *USENIX Security Symposium*, pp. 177–192. [84](#), [173](#), [178](#)
- P. Akrividis, C. Cadar, C. Raiciu, M. Costa, and M. Castro. 2008. Preventing memory error exploits with WIT. In *Proceedings of the 29th IEEE Symposium on Security and Privacy (S&P)*, pp. 263–277. DOI: [10.1109/SP.2008.30](https://doi.org/10.1109/SP.2008.30). [8](#), [58](#), [82](#), [84](#), [114](#), [173](#), [176](#), [178](#)
- P. Akrividis, M. Costa, M. Castro, and S. Hand. 2009. Baggy bounds checking: An efficient and backwards-compatible defense against out-of-bounds errors. In *USENIX Security Symposium*, pp. 51–66. [84](#), [173](#), [178](#)
- Aleph One. 1996. Smashing the stack for fun and profit. *Phrack*, 7. [11](#), [17](#)
- A. Alexandrov, P. Kmiec, and K. Schausser. 1999. Consh: Confined execution environment for Internet computations. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.488>. DOI: [10.1.1.57.488](https://doi.org/10.1.1.57.488). [16](#)
- G. Altekar and I. Stoica. 2010. Focus replay debugging effort on the control plane. In *USENIX Workshop on Hot Topics in Dependability*. [89](#)

- S. Andersen and V. Abella. August 2004. Changes to functionality in Windows XP service pack 2—part 3: Memory protection technologies. <http://technet.microsoft.com/en-us/library/bb457155.aspx>. 9, 19, 184
- J. Ansel. March 2014. Personal communication. 53
- J. Ansel, P. Marchenko, Ú. Erlingsson, E. Taylor, B. Chen, D. Schuff, D. Sehr, C. Biffle, and B. Yee. 2011. Language-independent sandboxing of just-in-time compilation and self-modifying code. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 355–366. DOI: [10.1145/1993316.1993540](https://doi.org/10.1145/1993316.1993540). 58, 59
- O. Arias, L. Davi, M. Hanreich, Y. Jin, P. Koeberl, D. Paul, A.-R. Sadeghi, and D. Sullivan. 2015. HAFIX: Hardware-assisted flow integrity extension. In *Proceedings of the 52nd Design Automation Conference (DAC)*, pp. 74:1–74:6. DOI: [10.1145/2744769.2744847](https://doi.org/10.1145/2744769.2744847). 182, 208, 209
- J.-P. Aumasson and D. J. Bernstein. 2012. SipHash: A fast short-input PRF. In *13th International Conference on Cryptology in India (INDOCRYPT)*. 73
- M. Backes, T. Holz, B. Kollenda, P. Koppe, S. Nürnberger, and J. Pewny. 2014. You can run but you can't read: Preventing disclosure exploits in executable code. In *ACM Conference on Computer and Communications Security (CCS)*. DOI: [10.1145/2660267.2660378](https://doi.org/10.1145/2660267.2660378), pp. 1342–1353. 65, 173, 177
- M. Backes and S. Nürnberger. 2014. Oxymoron: Making fine-grained memory randomization practical by allowing code sharing. In *23rd USENIX Security Symposium*, pp. 433–447. 64, 66
- A. Balasubramanian, M. S. Baranowski, A. Burtsev, and A. Panda. 2017. System programming in Rust: Beyond safety. In *Workshop on Hot Topics in Operating Systems (HotOS)*, pp. 94–99. DOI: [10.1145/3102980.3103006](https://doi.org/10.1145/3102980.3103006). 79
- C. Basile, Z. Kalbarczyk, and R. Iyer. 2002. A preemptive deterministic scheduling algorithm for multithreaded replicas. In *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 149–158. DOI: [10.1109/DSN.2003.1209926](https://doi.org/10.1109/DSN.2003.1209926). 230
- C. Basile, Z. Kalbarczyk, and R. K. Iyer. 2006. Active replication of multithreaded applications. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 17(5):448–465. DOI: [10.1109/TPDS.2006.56](https://doi.org/10.1109/TPDS.2006.56). 230
- A. Basu, J. Bobba, and M. D. Hill. 2011. Karma: Scalable deterministic record-replay. In *Proceedings of the International Conference on Supercomputing*, pp. 359–368. DOI: [10.1145/1995896.1995950](https://doi.org/10.1145/1995896.1995950). 230
- M. Bauer. 2006. Paranoid penguin: an introduction to Novell AppArmor. *Linux J.*, (148):13. 16
- A. Belay, A. Bittau, A. Mashtizadeh, D. Terei, D. Mazières, and C. Kozyrakis. 2012. Dune: Safe user-level access to privileged CPU features. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 335–348. 213

- T. Bergan, O. Anderson, J. Devietti, L. Ceze, and D. Grossman. 2010. CoreDet: A compiler and runtime system for deterministic multithreaded execution. *ACM SIGARCH Computer Architecture News*, 38(1):53–64. DOI: [10.1145/1735971.1736029](https://doi.org/10.1145/1735971.1736029). 230
- E. Berger, T. Yang, T. Liu, and G. Novark. 2009. Grace: Safe multithreaded programming for C/C++. *ACM Sigplan Notices*, 44(10):81–96. 230
- E. D. Berger and B. G. Zorn. 2006. DieHard: Probabilistic memory safety for unsafe languages. *ACM SIGPLAN Notices*, (6):158–168. DOI: [10.1145/1133255.1134000](https://doi.org/10.1145/1133255.1134000). 214
- E. Bhatkar, D. C. Duvarney, and R. Sekar. 2003. Address obfuscation: An efficient approach to combat a broad range of memory error exploits. In *Proceedings of the USENIX Security Symposium (SSYM)*, pp. 105–120. 10
- S. Bhatkar and R. Sekar. 2008. Data space randomization. In *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pp. 1–22. DOI: [10.1007/978-3-540-70542-0_1](https://doi.org/10.1007/978-3-540-70542-0_1). 11, 85
- S. Bhatkar, R. Sekar, and D. C. DuVarney. 2005. Efficient techniques for comprehensive protection from memory error exploits. In *Proceedings of the 14th USENIX Security Symposium (SSYM)*, pp. 17–17. <http://dl.acm.org/citation.cfm?id=1251398.1251415>. 10, 95
- D. Bigelow, T. Hobson, R. Rudd, W. Streilein, and H. Okhravi. 2015. Timely rerandomization for mitigating memory disclosures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 268–279. DOI: [10.1145/2810103.2813691](https://doi.org/10.1145/2810103.2813691). 11
- A. Bittau, A. Belay, A. Mashtizadeh, D. Mazieres, and D. Boneh. 2014. Hacking blind. In *Proceedings of the 35th IEEE Symposium on Security and Privacy (S&P)*, pp. 227–242. DOI: [10.1109/SP.2014.22](https://doi.org/10.1109/SP.2014.22). 62, 140, 141, 182, 239
- D. Blazakis. 2010. Interpreter exploitation. In *Proceedings of the 4th USENIX Conference on Offensive Technologies*, pp. 1–9. 59
- T. Bletsch, X. Jiang, and V. Freeh. 2011. Mitigating code-reuse attacks with control-flow locking. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 353–362. DOI: [10.1145/2076732.2076783](https://doi.org/10.1145/2076732.2076783). 208, 209
- T. Bletsch, X. Jiang, V. W. Freeh, and Z. Liang. 2011. Jump-oriented programming: A new class of code-reuse attack. In *Proceedings of the 6th ACM Symposium on Information, Computer, and Communications Security (ASIACCS)*, pp. 30–40. DOI: [10.1145/1966913.1966919](https://doi.org/10.1145/1966913.1966919). 20, 81, 82, 117
- E. Bosman and H. Bos. 2014. Framing signals—a return to portable shellcode. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 243–258. DOI: [10.1109/SP.2014.23](https://doi.org/10.1109/SP.2014.23). 31, 140
- K. Braden, S. Crane, L. Davi, M. Franz, P. Larsen, C. Liebchen, and A.-R. Sadeghi. 2016. Leakage-resilient layout randomization for mobile devices. In *23rd Annual Network and Distributed System Security Symposium (NDSS)*. 68

- S. Bratus, M. E. Locasto, M. L. Patterson, L. Sassaman, and A. Shubina. 2011. Exploit programming: From buffer overflows to “weird machines” and theory of computation. *Usenix ;login:* issue: December 2011, volume 36, number 6. 19
- E. Buchanan, R. Roemer, H. Shacham, and S. Savage. 2008. When good instructions go bad: Generalizing return-oriented programming to RISC. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, pp. 27–38. DOI: [10.1145/1455770.1455776](https://doi.org/10.1145/1455770.1455776). 233
- M. Budiu, Ú. Erlingsson, and M. Abadi. Architectural support for software-based protection. In *Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability (ASID)*, pp. 42–51. DOI: [10.1145/1181309.1181316](https://doi.org/10.1145/1181309.1181316). 208, 209
- N. Burow, S. A. Carr, S. Brunthaler, M. Payer, J. Nash, P. Larsen, and M. Franz. 2016. Control-flow integrity: Precision, security, and performance. *Computing Research Repository (CoRR)*. 50(1). <http://arxiv.org/abs/1602.04056>. 12, 28, 62, 82
- N. Burow, S. A. Carr, J. Nash, P. Larsen, M. Franz, S. Brunthaler, and M. Payer. 2017. Control-flow integrity: precision, security, and performance. *ACM Computing Surveys*. DOI: [10.1145/3054924](https://doi.org/10.1145/3054924). 12
- J. Butler and anonymous. 2004. Bypassing 3rd party Windows buffer overflow protection. *Phrack*, 11. 17
- N. Carlini, A. Barresi, M. Payer, D. Wagner, and T. R. Gross. 2015. Control-flow bending: On the effectiveness of control-flow integrity. In *Proceedings of the 24th USENIX Security Symposium*, pp. 161–176. <http://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/carlini>. 15, 20, 21, 59, 81, 82, 97, 137, 182, 183, 185, 186, 188, 200, 204, 211
- N. Carlini and D. Wagner. 2014. ROP is still dangerous: Breaking modern defenses. In *Proceedings of the 23rd USENIX Security Symposium*, pp. 385–399. <http://dl.acm.org/citation.cfm?id=2671225.2671250>. 15, 53, 82, 84, 86, 97, 114, 137, 139, 140, 176, 177, 179, 182, 183, 184, 186, 188, 200, 202, 209
- M. Castro, M. Costa, and T. Harris. 2006. Securing software by enforcing data-flow integrity. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 147–160. 11, 86
- M. Castro, M. Costa, J.-P. Martin, M. Peinado, P. Akritidis, A. Donnelly, P. Barham, and R. Black. 2009. Fast byte-granularity software fault isolation. In *ACM Symposium on Operating Systems Principles*, pp. 45–58. DOI: [10.1145/1629575.1629581](https://doi.org/10.1145/1629575.1629581). 86, 93
- L. Cavallaro. 2007. Comprehensive memory error protection via diversity and taint-tracking. PhD thesis, Università Degli Studi Di Milano. 214, 223, 237
- S. Checkoway, L. Davi, A. Dmitrienko, A.-R. Sadeghi, H. Shacham, and M. Winandy. 2010. Return-oriented programming without returns. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, pp. 559–572. DOI: [10.1145/1866307.1866370](https://doi.org/10.1145/1866307.1866370). 20, 81, 117, 183, 184, 185, 186, 200, 202

- S. Checkoway and H. Shacham. 2010. Escape from return-oriented programming: Return-oriented programming without returns (on the x86). Technical report CS2010-0954, UC San Diego. <http://cseweb.ucsd.edu/~hovav/dist/noret.pdf>. 200, 202
- P. Chen, Y. Fang, B. Mao, and L. Xie. 2011. JITDefender: A defense against JIT spraying attacks. In *26th IFIP International Information Security Conference*, volume 354, pp. 142–153. 60
- P. Chen, R. Wu, and B. Mao. 2013. JITSafe: A framework against just-in-time spraying attacks. *IET Information Security*, 7(4):283–292. DOI: [10.1049/iet-ifs.2012.0142](https://doi.org/10.1049/iet-ifs.2012.0142). 59, 60
- S. Chen, J. Xu, E. C. Sezer, P. Gauriar, and R. K. Iyer. 2005. Non-control-data attacks are realistic threats. In *Proceedings of the 14th USENIX Security Symposium*. <http://dl.acm.org/citation.cfm?id=1251398.1251410>. 21, 184
- X. Chen, A. Slowinska, D. Andriesse, H. Bos, and C. Giuffrida. 2015. StackArmor: Comprehensive protection from stack-based memory error vulnerabilities for binaries. In *Symposium on Network and Distributed System Security (NDSS)*. 173, 178
- Y. Chen, D. Zhang, R. Wang, R. Qiao, A. M. Azab, L. Lu, H. Vijayakumar, and W. Shen. 2017. NORAX: Enabling execute-only memory for COTS binaries on AArch64. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 304–319. DOI: [10.1109/SP.2017.30](https://doi.org/10.1109/SP.2017.30). 68
- Y. Cheng, Z. Zhou, M. Yu, X. Ding, and R. H. Deng. 2014. ROPecker: A generic and practical approach for defending against ROP attacks. In *Proceedings of the 21st Symposium on Network and Distributed System Security (NDSS)*. 117, 118, 119, 127, 173, 176, 182, 209
- M. Co, J. W. Davidson, J. D. Hiser, J. C. Knight, A. Nguyen-Tuong, W. Weimer, J. Burket, G. L. Frazier, T. M. Frazier, and B. Dutertre, et al. 2016. Double Helix and RAVEN: A system for cyber fault tolerance and recovery. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, p. 17. DOI: [10.1145/2897795.2897805](https://doi.org/10.1145/2897795.2897805). 214
- F. B. Cohen. 1993. Operating system protection through program evolution. *Computers & Security*, 12(6): 565–584. DOI: [10.1016/0167-4048\(93\)90054-9](https://doi.org/10.1016/0167-4048(93)90054-9). 62
- Corelan. 2011. Mona: A debugger plugin/exploit development Swiss army knife. <http://redmine.corelan.be/projects/mona>. 136
- C. Cowan, S. Beattie, G. Kroah-Hartman, C. Pu, P. Wagle, and V. Gligor. 2000. SubDomain: Parsimonious server security. In *Proceedings of the 14th USENIX Conference on System Administration*, pp. 355–368. 16
- C. Cowan, S. Beattie, J. Johansen, and P. Wagle. 2003. Pointguard™: Protecting pointers from buffer overflow vulnerabilities. In *Proceedings of the 12th USENIX Security Symposium (SSYM)*, pp. 7–7. <http://dl.acm.org/citation.cfm?id=1251353.1251360>. 11, 63, 76, 82, 85
- C. Cowan, C. Pu, D. Maier, H. Hinton, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang. 1998. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *Proceedings of the 7th USENIX Security Symposium*, volume 81, pp. 346–355. 61, 63, 82, 95, 211, 233

- B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser. 2006. N-variant systems: A secretless framework for security through diversity. In *Proceedings of the 15th USENIX Security Symposium*, 9. 211, 213, 214, 217, 237
- S. Crane, A. Homescu, and P. Larsen. 2016. Code randomization: Haven't we solved this problem yet? In *IEEE Cybersecurity Development (SecDev)*. DOI: 10.1109/SecDev.2016.036. 66
- S. Crane, P. Larsen, S. Brunthaler, and M. Franz. 2013. Booby trapping software. In *New Security Paradigms Workshop (NSPW)*, pp. 95–106. DOI: 10.1145/2535813.2535824. 68
- S. Crane, C. Liebchen, A. Homescu, L. Davi, P. Larsen, A.-R. Sadeghi, S. Brunthaler, and M. Franz. 2015. Readactor: Practical code randomization resilient to memory disclosure. In *36th IEEE Symposium on Security and Privacy (S&P)*, pp. 763–780. DOI: 10.1109/SP.2015.52. 11, 60, 66, 76, 173, 178
- S. Crane, S. Volckaert, F. Schuster, C. Liebchen, P. Larsen, L. Davi, A.-R. Sadeghi, T. Holz, B. De Sutter, and M. Franz. 2015. It's a TRaP: Table randomization and protection against function-reuse attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 243–255. DOI: 10.1145/2810103.2813682. 11, 68, 77, 159, 171, 173, 178
- J. Criswell, N. Dautenhahn, and V. Adve. 2014. KCoFI: Complete control-flow integrity for commodity operating system kernels. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 292–307. DOI: 10.1109/SP.2014.26. 58
- H. Cui, J. Simsa, Y.-H. Lin, H. Li, B. Blum, X. Xu, J. Yang, G. A. Gibson, and R. E. Bryant. 2013. Parrot: A practical runtime for deterministic, stable, and reliable threads. In *Proceedings of the 24th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 388–405. DOI: 10.1145/2517349.2522735. 230
- D. Dai Zovi. 2010. Practical return-oriented programming. Talk at *SOURCE Boston*, 2010. 117
- L. Dalessandro, D. Dice, M. Scott, N. Shavit, and M. Spear. 2010. Transactional mutex locks. In *Proceedings of the 16th International Euro-Par Conference on Parallel Processing: Part II*, pp. 2–13. 44
- T. H. Y. Dang, P. Maniatis, and D. Wagner. 2015. The performance cost of shadow stacks and stack canaries. In *Proceedings of the 10th ACM Symposium on Information, Computer, and Communications Security (ASIACCS)*, pp. 555–566. DOI: 10.1145/2714576.2714635. 10, 137, 208
- DarkReading. November 2009. Heap spraying: Attackers' latest weapon of choice. <http://www.darkreading.com/security/vulnerabilities/showArticle.jhtml?articleID=221901428>. 133
- L. Davi, A. Dmitrienko, M. Egele, T. Fischer, T. Holz, R. Hund, S. Nurnberger, and A.-R. Sadeghi. 2012. MoCFI: A framework to mitigate control-flow attacks on smartphones. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS)*. 58, 208

- L. Davi, P. Koeberl, and A.-R. Sadeghi. 2014. Hardware-assisted fine-grained control-flow integrity: Towards efficient protection of embedded systems against software exploitation. In *Annual Design Automation Conference—Special Session: Trusted Mobile Embedded Computing (DAC)*, pp. 1–6. DOI: [10.1145/2593069.2596656](https://doi.org/10.1145/2593069.2596656). 173, 174, 209
- L. Davi, D. Lehmann, A.-R. Sadeghi, and F. Monrose. 2014. Stitching the gadgets: On the ineffectiveness of coarse-grained control-flow integrity protection. In *Proceedings of the 23rd USENIX Security Symposium*, pp. 401–416. <http://dl.acm.org/citation.cfm?id=2671225.2671251>. 15, 43, 53, 82, 84, 86, 97, 114, 139, 140, 169, 174, 176, 177, 179, 182, 183, 184, 186, 188, 200, 209, 211
- L. Davi, C. Liebchen, A.-R. Sadeghi, K. Z. Snow, and F. Monrose. 2015. Isomeron: Code randomization resilient to (just-in-time) return-oriented programming. In *22nd Annual Network and Distributed System Security Symposium (NDSS)*. DOI: [10.14722/ndss.2015.23262](https://doi.org/10.14722/ndss.2015.23262). 64
- L. Davi, A.-R. Sadeghi, and M. Winandy. 2011. ROPdefender: A detection tool to defend against return-oriented programming attacks. In *ACM Symposium on Information, Computer, and Communications Security (ASIACCS)*, pp. 40–51. DOI: [10.1145/1966913.1966920](https://doi.org/10.1145/1966913.1966920). 139, 141
- L. de Moura and N. Bjørner. 2009. Generalized, efficient array decision procedures. In *Formal Methods in Computer Aided Design (FMCAD)*. DOI: [10.1109/FMCAD.2009.5351142](https://doi.org/10.1109/FMCAD.2009.5351142). 161
- L. M. de Moura and N. Bjørner. 2008. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 337–340. 140, 161
- T. de Raadt. 2005. Exploit mitigation techniques. <http://www.openbsd.org/papers/ven05-deraadt/index.html>. 8
- J. Dean, D. Grove, and C. Chambers. 1995. Optimization of object-oriented programs using static class hierarchy analysis. In *European Conference on Object-Oriented Programming (ECOOP)*, pp. 77–101. 32
- D. Dechev. 2011. The ABA problem in multicore data structures with collaborating operations. In *7th International Conference on Collaborative Computing: Networking, Applications, and Worksharing (CollaborateCom)*, pp. 158–167. DOI: [10.4108/icst.collaboratecom.2011.247161](https://doi.org/10.4108/icst.collaboratecom.2011.247161). 44
- L. Deng, Q. Zeng, and Y. Liu. 2015. ISboxing: An instruction substitution based data sandboxing for x86 untrusted libraries. In *30th International Conference on ICT Systems Security and Privacy Protection*, pp. 386–400. 41
- L. P. Deutsch and A. M. Schiffman. 1984. Efficient implementation of the Smalltalk-80 system. In *Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pp. 297–302. DOI: [10.1145/800017.800542](https://doi.org/10.1145/800017.800542). 54
- J. Devietti, C. Blundell, M. M. K. Martin, and S. Zdancewic. 2008. HardBound: Architectural support for spatial safety of the C programming language. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 103–114. DOI: [10.1145/1353534.1346295](https://doi.org/10.1145/1353534.1346295). 109

- J. Devietti, B. Lucia, L. Ceze, and M. Oskin. 2009. DMP: Deterministic shared memory multiprocessing. *ACM SIGARCH Computer Architecture News*, 37(1):85–96. DOI: [10.1145/1508244.1508255](https://doi.org/10.1145/1508244.1508255). 230
- D. Dewey and J. T. Giffin. 2012. Static detection of C++ vtable escape vulnerabilities in binary code. In *Symposium on Network and Distributed System Security (NDSS)*. 171
- D. Dhurjati, S. Kowshik, and V. Adve. June 2006. SAFECode: Enforcing alias analysis for weakly typed languages. *SIGPLAN Notices*, 41 (6): 144–157. DOI: [10.1145/1133255.1133999](https://doi.org/10.1145/1133255.1133999). 82, 84
- U. Drepper. April 2006. SELinux memory protection tests. <http://www.akkadia.org/drepper/selinux-mem.html>. 238
- V. D'Silva, M. Payer, and D. Song. 2015. The Correctness-Security Gap in Compiler Optimization. In *LangSec'15: Second Workshop on Language-Theoretic Security*. DOI: [10.1109/SPW.2015.33](https://doi.org/10.1109/SPW.2015.33). 16
- T. Durden. 2002. Bypassing PaX ASLR protection. *Phrack*, 11. 10, 17
- EEMBC. The embedded microprocessor benchmark consortium: EEMBC benchmark suite. <http://www.eembc.org>. 206
- Ú Erlingsson, M. Abadi, M. Vrabie, M. Budiu, and G. Necula. 2006. XFI: Software guards for system address spaces. In *Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation*, pp. 75–88. 58, 86, 95
- H. Etoh and K. Yoda. June 2000. Protecting from stack-smashing attacks. Technical report, IBM Research Division, Tokyo Research Laboratory. 63
- C. Evans. 2013. Exploiting 64-bit Linux like a boss. <http://scarybeastsecurity.blogspot.com/2013/02/exploiting-64-bit-linux-like-boss.html>. 117
- I. Evans, S. Fingeret, J. Gonzalez, U. Otgonbaatar, T. Tang, H. E. Shrobe, S. Sidiroglou-Douskos, M. Rinard, and H. Okhravi. 2015. Missing the point(er): On the effectiveness of code pointer integrity. In *36th IEEE Symposium on Security and Privacy, (S&P)*, pp. 781–796. DOI: [10.1109/SP.2015.53](https://doi.org/10.1109/SP.2015.53). 11, 62, 87
- I. Evans, F. Long, U. Otgonbaatar, H. Shrobe, M. Rinard, H. Okhravi, and S. Sidiroglou-Douskos. 2015. Control jujutsu: On the weaknesses of fine-grained control flow integrity. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 901–913. DOI: [10.1145/2810103.2813646](https://doi.org/10.1145/2810103.2813646). 20, 21, 59, 82, 97, 137, 211
- Federal Communications Commission. 2014. Measuring broadband America—2014. <http://www.fcc.gov/reports/measuring-broadband-america-2014>. 256
- C. Fetzer and M. Suesskraut. 2008. SwitchBlade: Enforcing dynamic personalized system call models. In *Proceedings of the 3rd European Conference on Computer Systems*, pp. 273–286. DOI: [10.1145/1357010.1352621](https://doi.org/10.1145/1357010.1352621). 16
- A. Fokin, E. Derevenetc, A. Chernov, and K. Troshina. 2011. SmartDec: Approaching C++ decompilation. In *Working Conference on Reverse Engineering (WCRE)*. 171
- B. Ford and R. Cox. 2008. Vx32: Lightweight user-level sandboxing on the x86. In *Proceedings of the USENIX ATC*, pp. 293–306. 8, 9

- M. Frantzen and M. Shuey. 2001. StackGhost: Hardware facilitated stack protection. In *USENIX Security Symposium*. 139, 141
- I. Fratric. 2012. Runtime prevention of return-oriented programming attacks. <http://github.com/ivanfratric/ropguard/blob/master/doc/ropguard.pdf>. 139, 173, 176
- Gaisler Research. LEON3 synthesizable processor. <http://www.gaisler.com>. 183, 206
- A. Gal, B. Eich, M. Shaver, D. Anderson, D. Mandelin, M. R. Haghighat, B. Kaplan, G. Hoare, B. Zbarsky, J. Orendorff, J. Ruderman, E. W. Smith, R. Reitmaier, M. Bebenita, M. Chang, and M. Franz. 2009. Trace-based just-in-time type specialization for dynamic languages. In *Proceedings ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 465–478. DOI: 10.1145/1543135.1542528. 50
- T. Garfinkel, B. Pfaff, and M. Rosenblum. 2004. Ostia: A delegating architecture for secure system call interposition. In *Network and Distributed System Security Symposium (NDSS)*. 241
- R. Gawlik and T. Holz. 2014. Towards automated integrity protection of C++ virtual function tables in binary programs. In *Annual Computer Security Applications Conference (ACSAC)*, pp. 396–405. 171, 173, 176, 182
- R. Gawlik, B. Kollenda, P. Koppe, B. Garmany, and T. Holz. 2016. Enabling client-side crash-resistance to overcome diversification and information hiding. In *23rd Annual Network and Distributed System Security Symposium (NDSS)*. 68
- X. Ge, M. Payer, and T. Jaeger. 2017. An evil copy: How the loader betrays you. In *Network and Distributed System Security Symposium (NDSS)*. DOI: 10.14722/ndss.2017.23199 . 15
- J. Gionta, W. Enck, and P. Ning. 2015. HideM: Protecting the contents of userspace memory in the face of disclosure vulnerabilities. In *5th ACM Conference on Data and Application Security and Privacy (CODASPY)*, pp. 325–336. DOI: 10.1145/2699026.2699107. 65
- GNU.org. The GNU C library: Environment access. http://www.gnu.org/software/libc/manual/html_node/Environment-Access.html. 220
- E. Göktas, E. Athanasopoulos, H. Bos, and G. Portokalidis. 2014a. Out of control: Overcoming control-flow integrity. In *Proceedings of the 35th IEEE Symposium on Security and Privacy (S&P)*, pp. 575–589. DOI: 10.1109/SP.2014.43. 15, 53, 82, 84, 86, 97, 114, 124, 125, 126, 129, 134, 136, 137, 139, 140, 174, 175, 177, 182, 183, 186, 188, 200, 202, 211
- E. Göktas, E. Athanasopoulos, M. Polychronakis, H. Bos, and G. Portokalidis. 2014b. Size does matter: Why using gadget-chain length to prevent code-reuse attacks is hard. In *Proceedings of the 23rd USENIX Security Symposium*. <http://dl.acm.org/citation.cfm?id=2671225.2671252>. 122, 139, 140, 169, 177, 179, 182, 186, 188, 209
- E. Göktas, R. Gawlik, B. Kollenda, E. Athanasopoulos, G. Portokalidis, C. Giuffrida, and H. Bos. 2016. Undermining information hiding (and what to do about it). In *Proceedings of the 25th USENIX Security Symposium*, pp. 105–119. 11, 68
- I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer. 1996. A secure environment for untrusted helper applications: Confining the wily hacker. In *Proceedings of the 6th USENIX Security Symposium (SSYM)*. 16

- Google Chromium Project. 2013. Undefined behavior sanitizer. <http://www.chromium.org/developers/testing/undefinedbehaviorsanitizer>. 7
- B. Gras, K. Razavi, E. Bosman, H. Bos, and C. Giuffrida. 2017. ASLR on the line: Practical cache attacks on the MMU. In *Annual Network and Distributed System Security Symposium (NDSS)*. 67
- Y. Guillot and A. Gazet. 2010. Automatic binary deobfuscation. *J. Comput. Virol.* 6(3): pp. 261–276. DOI: [10.1007/s11416-009-0126-4](https://doi.org/10.1007/s11416-009-0126-4). 160
- I. Haller, E. Göktas, E. Athanasopoulos, G. Portokalidis, and H. Bos. 2015. ShrinkWrap: VTable protection without loose ends. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pp. 341–350. DOI: [10.1145/2818000.2818025](https://doi.org/10.1145/2818000.2818025). 136
- I. Haller, Y. Jeon, H. Peng, M. Payer, H. Bos, C. Giuffrida, and E. van der Kouwe. 2016. TypeSanitizer: Practical type confusion detection. In *ACM Conference on Computer and Communication Security (CCS)*. DOI: [10.1145/2976749.2978405](https://doi.org/10.1145/2976749.2978405). 7
- N. Hasabnis, A. Misra, and R. Sekar. 2012. Light-weight bounds checking. In *IEEE/ACM Symposium on Code Generation and Optimization*. DOI: [10.1145/2259016.2259034](https://doi.org/10.1145/2259016.2259034). 84
- Hex-Rays. 2017. IDA Pro. <http://www.hex-rays.com/index.shtml>. 128
- M. Hicks. 2014. What is memory safety? <http://www.pl-enthusiast.net/2014/07/21/memory-safety/>. 4
- E. Hiroaki and Y. Kunikazu. 2001. ProPolice: Improved stack-smashing attack detection. *IPSJ SIG Notes*, pp. 181–188. 11
- J. Hiser, A. Nguyen, M. Co, M. Hall, and J. W. Davidson. 2012. ILR: Where'd my gadgets go? In *33rd IEEE Symposium on Security and Privacy (S&P)*, pp. 571–585. DOI: [10.1109/SP.2012.39](https://doi.org/10.1109/SP.2012.39). 11, 66
- J. D. Hiser, D. Williams, A. Filipi, J. W. Davidson, and B. R. Childers. 2006. Evaluating fragment construction policies for SDT systems. In *Proceedings of the 2nd International Conference on Virtual Execution Environments (VEE)*, pp. 122–132. DOI: [10.1145/1134760.1134778](https://doi.org/10.1145/1134760.1134778). 8, 9
- U. Hölzle, C. Chambers, and D. Ungar. 1991. Optimizing dynamically-typed object-oriented languages with polymorphic inline caches. In *European Conference on Object-Oriented Programming (ECOOP)*, pp. 21–38. 54
- U. Hölzle, C. Chambers, and D. Ungar. 1992. Debugging optimized code with dynamic deoptimization. In *Proceedings ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 32–43. DOI: [10.1145/143103.143114](https://doi.org/10.1145/143103.143114). 54
- A. Homescu, S. Brunthaler, P. Larsen, and M. Franz. 2013. Librando: Transparent code randomization for just-in-time compilers. *CCS '13*, pp. 993–1004. DOI: [10.1145/2508859.2516675](https://doi.org/10.1145/2508859.2516675). 58, 59
- A. Homescu, S. Neisius, P. Larsen, S. Brunthaler, and M. Franz. 2013. Profile-guided automated software diversity. In *IEEE/ACM Symposium on Code Generation and Optimization*, pp. 1–11. DOI: [10.1109/CGO.2013.6494997](https://doi.org/10.1109/CGO.2013.6494997). 85

- P. Hosek and C. Cadar. 2013. Safe software updates via multi-version execution. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE'13)*, pp. 612–621. DOI: [10.1109/ICSE.2013.6606607](https://doi.org/10.1109/ICSE.2013.6606607). 214, 256, 257
- Petr Hosek and Cristian Cadar. 2015. Varan the unbelievable: An efficient n-version execution framework. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 339–353. DOI: [10.1145/2694344.2694390](https://doi.org/10.1145/2694344.2694390). 214, 215, 218, 224, 227, 256, 257
- H. Hu, Z. L. Chua, S. Adrian, P. Saxena, and Z. Liang. 2015. Automatic generation of data-oriented exploits. In *24th USENIX Security Symposium*, pp. 177–192. <http://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/hu.21>
- R. Hund, C. Willems, and T. Holz. 2013. Practical timing side channel attacks against kernel space ASLR. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 191–205. DOI: [10.1109/SP.2013.23](https://doi.org/10.1109/SP.2013.23). 82, 86, 141
- G. Hunt and D. Brubacher. 1999. Detours: Binary interception of win32 functions. In *Usenix Windows NT Symposium*, pp. 135–143. 232
- Intel. 2013. *Intel Architecture Instruction Set Extensions Programming Reference*. <http://download-software.intel.com/sites/default/files/319433-015.pdf>. 108
- Intel. 2013. Introduction to Intel memory protection extensions. <http://software.intel.com/en-us/articles/introduction-to-intel-memory-protection-extensions>. 93
- Intel. 2013. *Intel 64 and IA-32 Architectures Software Developer's Manual—Combined Volumes 1, 2a, 2b, 2c, 3a, 3b, and 3c*. 178
- Intel. 2014. *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2: Instruction Set Reference, A–Z*. 2014. 223, 224
- Itanium C++ ABI. <http://mentorembedded.github.io/cxx-abi/abi.html>. 32
- A. Jaleel. 2007. Memory characterization of workloads using instrumentation-driven simulation—a pin-based memory characterization of the SPEC CPU2000 and SPEC CPU2006 benchmark suites. *technical report*. <http://www.glue.umd.edu/~ajaleel/workload/>. 253
- D. Jang, Z. Tatlock, and S. Lerner. 2014. SAFEDISPATCH: Securing C++ virtual calls from memory corruption attacks. In *Symposium on Network and Distributed System Security (NDSS)*. 32, 173, 176
- jduck. 2010. The latest Adobe exploit and session upgrading. <http://bugix-security.blogspot.de/2010/03/adobe-pdf-libtiff-working-exploiteve.html>. 182
- T. Jim, J. G. Morrisett, D. Grossman, M. W. Hicks, J. Cheney, and Y. Wang. 2002. Cyclone: A safe dialect of C. In *USENIX Annual Technical Conference*. 5, 82, 84, 88, 95
- N. Joly. 2013. Advanced exploitation of Internet Explorer 10/Windows 8 overflow (Pwn2Own 2013). http://www.vupen.com/blog/20130522.Advanced_Exploitation_of_IE10_Windows8_Pwn2Own_2013.php. 117, 124, 162
- M. Kayaalp, M. Ozsoy, N. Abu-Ghazaleh, and D. Ponomarev. 2012. Branch regulation: Low-overhead protection from code reuse attacks. In *Proceedings of the 39th Annual*

- International Symposium on Computer Architecture (ISCA)*. <http://dl.acm.org/citation.cfm?id=2337159.2337171>. DOI: [10.1109/ISCA.2012.6237009](https://doi.org/10.1109/ISCA.2012.6237009). 182, 209
- M. Kayaalp, T. Schmitt, J. Nomani, D. Ponomarev, and N. Abu-Ghazaleh. 2013. Scrap: Architecture for signature-based protection from code reuse attacks. In *IEEE 19th International Symposium on High Performance Computer Architecture (HPCA2013)*, pp. 258–269. DOI: [10.1109/HPCA.2013.6522324](https://doi.org/10.1109/HPCA.2013.6522324). 209
- C. Kil, J. Jun, C. Bookholt, J. Xu, and P. Ning. 2006. Address space layout permutation (ASLP): Towards fine-grained randomization of commodity software. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC)*, pp. 339–348. DOI: [10.1109/ACSAC.2006.9](https://doi.org/10.1109/ACSAC.2006.9). 11, 85
- V. Kiriansky, D. Bruening, and S. P. Amarasinghe. 2002. Secure execution via program shepherding. In *Proceedings 11th USENIX Security Symposium*, pp. 191–206. 8, 9
- K. Koning, H. Bos, and C. Giuffrida. 2016. Secure and efficient multi-variant execution using hardware-assisted process virtualization. In *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 431–442. DOI: [10.1109/DSN.2016.46](https://doi.org/10.1109/DSN.2016.46). 211, 214, 217
- T. Kornau. 2010. Return oriented programming for the ARM architecture. Ph.D. thesis, Master's thesis, Ruhr-Universität Bochum. 233
- V. Kuznetsov, L. Szekeres, M. Payer, G. Candea, R. Sekar, and D. Song. 2014a. Code-pointer integrity. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 147–163. 9, 10, 59, 62, 105, 106, 107, 173, 178, 179
- V. Kuznetsov, L. Szekeres, M. Payer, G. Candea, R. Sekar, and D. Song. 2014b. Code-Pointer Integrity website. <http://dslab.epfl.ch/proj/cpi/>. 179
- V. Kuznetsov, L. Szekeres, M. Payer, G. Candea, and D. Song. 2015. Poster: Getting the point (er): On the feasibility of attacks on code-pointer integrity. In *36th IEEE Symposium on Security and Privacy (S&P)*. 87
- P. Larsen, A. Homescu, S. Brunthaler, and M. Franz. 2014. SoK: Automated software diversity. In *Proceedings of the 35th IEEE Symposium on Security and Privacy (S&P)*, pp. 276–291. DOI: [10.1109/SP.2014.25](https://doi.org/10.1109/SP.2014.25). 11, 62, 66, 250, 252
- C. Lattner and V. Adve. 2005. Automatic pool allocation: Improving performance by controlling data structure layout in the heap. In *ACM Conference on Programming Language Design and Implementation*, pp. 129–142. DOI: [10.1145/1064978.1065027](https://doi.org/10.1145/1064978.1065027). 91, 108
- C. Lattner, A. Lenharth, and V. Adve. 2007. Making context-sensitive points-to analysis with heap cloning practical for the real world. In *ACM Conference on Programming Language Design and Implementation*, pp. 278–289. DOI: [10.1145/1273442.1250766](https://doi.org/10.1145/1273442.1250766). 91, 108
- B. Lee, C. Song, T. Kim, and W. Lee. 2015. Type casting verification: Stopping an emerging attack vector. In *USENIX Security 15*, pp. 81–96. <http://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/lee>. 7

- D. Lee, B. Wester, K. Veeraraghavan, S. Narayanasamy, P. M. Chen, and J. Flinn. 2010. Respec: Efficient online multiprocessor replay via speculation and external determinism. *ACM SIGARCH Computer Architecture News*, 38(1):77–90. DOI: [10.1145/1736020.1736031](https://doi.org/10.1145/1736020.1736031). 230
- J. Lettner, B. Kollenda, A. Homescu, P. Larsen, F. Schuster, L. Davi, A.-R. Sadeghi, T. Holz, and M. Franz. 2016. Subversive-C: Abusing and protecting dynamic message dispatch. In *USENIX Annual Technical Conference (ATC)*, pp. 209–221. 70, 140
- E. Levy. 1996. Smashing the stack for fun and profit. *Phrack*, 7. 61
- J. Li, Z. Wang, T. K. Bletsch, D. Srinivasan, M. C. Grace, and X. Jiang. 2011. Comprehensive and efficient protection of kernel control data. *IEEE Transactions on Information Forensics and Security*, 6(4):1404–1417. DOI: [10.1109/TIFS.2011.2159712](https://doi.org/10.1109/TIFS.2011.2159712). 82, 86
- C. Liebchen, M. Negro, P. Larsen, L. Davi, A.-R. Sadeghi, S. Crane, M. Qunaibit, M. Franz, and M. Conti. 2015. Losing control: On the effectiveness of control-flow integrity under stack attacks. In *ACM Conference on Computer and Communications Security (CCS)*. DOI: [10.1145/2810103.2813671](https://doi.org/10.1145/2810103.2813671). 182, 183, 205
- Linux Man-Pages Project. 2017a. tc-netem(8)—Linux manual page. 256
- Linux Man-Pages Project. 2017b. shmop(2)—Linux manual page. 247
- Linux Programmer's Manual*. 2017a. vdso(7)—Linux manual page. 223
- Linux Programmer's Manual*. 2017b. getauxval(3)—Linux manual page. 224
- Linux Programmer's Manual*. 2017c. signal(7)—Linux manual page. 225
- T. Liu, C. Curtsinger, and E. Berger. 2011. DTHREADS: Efficient deterministic multithreading. In *Proceedings of the 23rd ACM Symposium on Operating System Principles (SOSP)*, pp. 327–336. DOI: [10.1145/2043556.2043587](https://doi.org/10.1145/2043556.2043587). 230
- LLVM. The LLVM compiler infrastructure. <http://llvm.org/>. 102
- K. Lu, X. Zhou, T. Bergan, and X. Wang. 2014. Efficient deterministic multithreading without global barriers. In *Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pp. 287–300. DOI: [10.1145/2555243.2555252](https://doi.org/10.1145/2555243.2555252). 230
- K. Lu, C. Song, B. Lee, S. P. Chung, T. Kim, and W. Lee. 2015. ASLR-Guard: Stopping address space leakage for code reuse attacks. In *ACM Conference on Computer and Communications Security (CCS)*, pp. 280–291. DOI: [10.1145/2810103.2813694](https://doi.org/10.1145/2810103.2813694). 68
- J. Maebe, M. Ronsse, and K. D. Bosschere. 2003. Instrumenting JVMs at the machine code level. In *3rd PA3CT symposium*, volume 19, pp. 105–107. 222
- G. Maisuradze, M. Backes, and C. Rossow. 2003. What cannot be read, cannot be leveraged? Revisiting assumptions of JIT-ROP defenses. In *USENIX Security Symposium*. 67
- M. Marschalek. 2014. Dig deeper into the IE vulnerability (cve-2014-1776) exploit. <http://www.cyphort.com/dig-deeper-ie-vulnerability-cve-2014-1776-exploit/>. 182
- A. J. Mashtizadeh, A. Bittau, D. Mazieres, and D. Boneh. 2014. Cryptographically enforced control flow integrity. <http://arxiv.org/abs/1408.1451>. 86

- A. J. Mashtizadeh, A. Bittau, D. Boneh, and D. Mazières. 2015. CCFI: Cryptographically enforced control flow integrity. In *ACM Conference on Computer and Communications Security (CCS)*, pp. 941–951. DOI: [10.1145/2810103.2813676](https://doi.org/10.1145/2810103.2813676). 72, 76, 77
- M. Matz, J. Hubicka, A. Jaeger, and M. Mitchell. 2013. System V application binary interface: AMD64 architecture processor supplement. <http://x86-64.org/documentation/abi.pdf>. 150
- M. Maurer and D. Brumley. 2012. Tachyon: Tandem execution for efficient live patch testing. In *USENIX Security Symposium*, pp. 617–630. 214, 256, 257
- S. McCamant and G. Morrisett. 2006. Evaluating SFI for a CISC architecture. In *Proceedings of the 15th USENIX Security Symposium*. 41, 59, 68, 86
- H. Meer. 2010. Memory corruption attacks: The (almost) complete history. In *Proceedings of Blackhat USA*. 62
- T. Merrifield and J. Eriksson. 2013. Conversion: Multi-version concurrency control for main memory segments. In *Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys)*, pp. 127–139. DOI: [10.1145/2465351.2465365](https://doi.org/10.1145/2465351.2465365). 230
- Microsoft Corp. November 2014. Enhanced mitigation experience toolkit (EMET) 5.1. <http://technet.microsoft.com/en-us/security/jj653751>. 173, 176
- Microsoft Developer Network. 2017. Argument passing and naming conventions. <http://msdn.microsoft.com/en-us/library/984x0h58.aspx>. 149, 151, 154
- V. Mohan, P. Larsen, S. Brunthaler, K. W. Hamlen, and M. Franz. 2015. Opaque control-flow integrity. In *Proceedings of the 22nd Annual Network and Distributed System Security Symposium (NDSS)*. <http://www.internetsociety.org/doc/opaque-control-flow-integrity>. 173, 177, 182
- J. R. Moser. 2006. Virtual machines and memory protections. <http://lwn.net/Articles/210272/>. 238
- G. Murphy. 2012. Position independent executables—adoption recommendations for packages. <http://people.redhat.com/~gmurphy/files/pie.odt>. 238
- S. Nagarakatte, M. M. K. Martin, and S. Zdancewic. 2012. Watchdog: Hardware for safe and secure manual memory management and full memory safety. In *International Symposium on Computer Architecture*, pp. 189–200. DOI: [10.1145/2366231.2337181](https://doi.org/10.1145/2366231.2337181). 109
- S. Nagarakatte, M. M. K. Martin, and S. Zdancewic. 2015. Everything you want to know about pointer-based checking. In *First Summit on Advances in Programming Languages (SNAPL)*. DOI: [10.4230/LIPIcs.SNAPL.2015.190](https://doi.org/10.4230/LIPIcs.SNAPL.2015.190). 5
- S. Nagarakatte, J. Zhao, M. M. K. Martin, and S. Zdancewic. 2009. SoftBound: Highly compatible and complete spatial memory safety for C. In *ACM Sigplan Notices*, volume 44, pp. 245–258. DOI: [10.1145/1542476.1542504](https://doi.org/10.1145/1542476.1542504). 4, 5, 36, 82, 84, 88, 91, 97, 99, 101, 102, 110, 112, 211
- S. Nagarakatte, J. Zhao, M. M. K. Martin, and S. Zdancewic. 2010. CETS: Compiler enforced temporal safety for C. In *ACM Sigplan Notices*, volume 45, pp. 31–40. DOI: [10.1145/1806651.1806657](https://doi.org/10.1145/1806651.1806657). 6, 83, 84, 88, 89, 91, 98, 108, 173, 178, 211

- G. C. Necula, J. Condit, M. Harren, S. McPeak, and W. Weimer. 2005. CCured: Type-safe retrofitting of legacy software. *ACM Transactions on Programming Languages and Systems*, 27(3):477–526. DOI: [10.1145/1065887.1065892](https://doi.org/10.1145/1065887.1065892). 5, 82, 84, 88, 95
- Nergal. December 2001. The advanced return-into-lib(c) exploits (PaX case study). *Phrack*, 58 (4): 54. [http://www.phrack.org/archives/58/p58_0x04_Advanced%20return-into-lib\(c\)%20exploits%20\(PaX%20case%20study\)_by_nergal.txt](http://www.phrack.org/archives/58/p58_0x04_Advanced%20return-into-lib(c)%20exploits%20(PaX%20case%20study)_by_nergal.txt). 81, 82, 185, 203
- B. Niu. 2015. Practical control-flow integrity. Ph.D. thesis, Lehigh University. 26, 37, 39, 56, 60
- B. Niu and G. Tan. 2013. Monitor integrity protection with space efficiency and separate compilation. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 199–210. DOI: [10.1145/2508859.2516649](https://doi.org/10.1145/2508859.2516649). 39, 82, 86, 173, 175
- B. Niu and G. Tan. 2014a. Modular control-flow integrity. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. DOI: [10.1145/2594291.2594295](https://doi.org/10.1145/2594291.2594295). 26, 27, 40, 44, 58, 82, 110, 114, 182
- B. Niu and G. Tan. 2014b. RockJIT: Securing just-in-time compilation using modular control-flow integrity. In *ACM Conference on Computer and Communication Security (CCS)*, pp. 1317–1328. DOI: [10.1145/2660267.2660281](https://doi.org/10.1145/2660267.2660281). 9, 26, 34
- B. Niu and G. Tan. 2015. Per-input control-flow integrity. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 914–926. DOI: [10.1145/2810103.2813644](https://doi.org/10.1145/2810103.2813644). 14, 30, 59
- A. Oikonomopoulos, E. Athanasopoulos, H. Bos, and C. Giuffrida. 2016. Poking holes in information hiding. In *25th USENIX Security Symposium*, pp. 121–138. 11, 68, 94
- M. Olszewski, J. Ansel, and S. Amarasinghe. 2009. Kendo: Efficient deterministic multithreading in software. *ACM Sigplan Notices*, 44(3):97–108. DOI: [10.1145/1508244.1508256](https://doi.org/10.1145/1508244.1508256). 230
- K. Onarlioglu, L. Bilge, A. Lanzi, D. Balzarotti, and E. Kirda. 2010. G-Free: Defeating return-oriented programming through gadget-less binaries. In *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC)*, pp. 49–58. DOI: [10.1145/1920261.1920269](https://doi.org/10.1145/1920261.1920269). 173, 177, 210
- V. Pappas, M. Polychronakis, and A. D. Keromytis. 2013. Transparent ROP exploit mitigation using indirect branch tracing. In *Proceedings of the 22nd USENIX Security Symposium*, pp. 447–462. 117, 118, 119, 127, 173, 176, 182, 209
- A. Pawlowski, M. Contag, V. van der Veen, C. Ouwehand, Thorsten Holz, Herbert Bos, Elias Athanasopoulos, and Cristiano Giuffrida. 2017. Marx: Uncovering class hierarchies in C++ programs. In *Annual Network and Distributed System Security Symposium (NDSS)*. 67
- PaX Team. 2004a. Address space layout randomization. <http://pax.grsecurity.net/docs/aslr.txt>, 2004a. 82, 85, 211
- PaX Team. 2004b PaX non-executable pp. design & implementation. <http://pax.grsecurity.net/docs/noexec.txt>, 2004b. 8, 211

- M. Payer. 2012. Safe loading and efficient runtime confinement: A foundation for secure execution. Ph.D. thesis, ETH Zurich. <http://nebelwelt.net/publications/12PhD>. DOI: [10.1109/SP.2012.11](https://doi.org/10.1109/SP.2012.11). 8
- M. Payer, A. Barresi, and T. R. Gross. 2015. Fine-grained control-flow integrity through binary hardening. In *Proceedings of the 12th Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*. DOI: [10.1007/978-3-319-20550-2_8](https://doi.org/10.1007/978-3-319-20550-2_8). 10, 14, 58, 173, 174
- M. Payer and T. R. Gross. 2011. Fine-grained user-space security through virtualization. In *Proceedings of the 7th International Conference on Virtual Execution Environments (VEE)*. DOI: [10.1145/1952682.1952703](https://doi.org/10.1145/1952682.1952703). 8, 9
- A. Pelletier. 2012. Advanced exploitation of Internet Explorer heap overflow (Pwn2Own 2012 exploit). VUPEN Vulnerability Research Team (VRT) blog. http://www.vupen.com/blog/20120710.Advanced_Exploitation_of_Internet_Explorer_HeapOv_CVE-2012-1876.php. 124, 131
- J. Pewny and T. Holz. 2013. Control-flow restrictor: Compiler-based CFI for iOS. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pp. 309–318. DOI: [10.1145/2523649.2523674](https://doi.org/10.1145/2523649.2523674). 58
- Phoronix. Phoronix test suite. <http://www.phoronix-test-suite.com/>. 114
- A. Prakash, X. Hu, and H. Yin. 2015. vfGuard: Strict protection for virtual function calls in COTS C++ binaries. In *Symposium on Network and Distributed System Security (NDSS)*. 58, 160, 170, 171, 173, 176, 182
- N. Provos. 2003. Improving host security with system call policies. In *Proceedings of the 12th USENIX Security Symposium (SSYM)*, volume 12, pp. 18–18. <http://dl.acm.org/citation.cfm?id=1251353.1251371>. 16, 241
- H. P. Reiser, J. Domaschka, F. J. Hauck, R. Kapitza, and W. Schröder-Preikschat. 2006. Consistent replication of multithreaded distributed objects. In *IEEE Symposium on Reliable Distributed Systems*, pp. 257–266. DOI: [10.1109/SRDS.2006.14](https://doi.org/10.1109/SRDS.2006.14). 230
- R. Roemer, E. Buchanan, H. Shacham, and S. Savage. 2012. Return-oriented programming: Systems, languages, and applications. *ACM Trans. Inf. Syst. Secur.*, 15(1):2:1–2:34. DOI: [10.1145/2133375.2133377](https://doi.org/10.1145/2133375.2133377). 20, 117, 181, 185
- R. Rudd, R. Skowrya, D. Bigelow, V. Dedhia, T. Hobson, S. Crane, C. Liebchen, P. Larsen, L. Davi, M. Franz, A.-R. Sadeghi, and H. Okhravi. 2017. Address oblivious code reuse: On the effectiveness of leakage resilient diversity. In *Annual Network and Distributed System Security Symposium (NDSS)*. 69
- J. M. Rushby. 1981. Design and verification of secure systems. In *Proceedings of the 8th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 12–21. DOI: [10.1145/800216.806586](https://doi.org/10.1145/800216.806586). 214
- M. Russinovich, D. A. Solomon, and A. Ionescu. 2012. *Windows Internals, Part 1*. Microsoft Press, 6th edition. ISBN 978-0-7356-4873-9. 155, 175
- SafeStack. Clang documentation: Safestack. <http://clang.llvm.org/docs/SafeStack.html>. 102

- B. Salamat. 2009. Multi-variant execution: Run-time defense against malicious code injection attacks. Ph.D. thesis, University of California at Irvine. 218
- B. Salamat, T. Jackson, A. Gal, and M. Franz. 2009. Orchestra: Intrusion detection using parallel execution and monitoring of program variants in user-space. In *Proceedings of the 4th ACM European Conference on Computer Systems (EuroSys)*, pp. 33–46. DOI: 10.1145/1519065.1519071. 211, 213, 214, 217, 227, 256, 257
- J. Salwan. 2011. ROPGadget. <http://shell-storm.org/project/ROPgadget/>. 136
- F. Schuster. July 2015. *Securing Application Software in Modern Adversarial Settings*. Ph.D. thesis, Katholieke Universiteit Leuven. 140
- F. Schuster, T. Tendyck, C. Liebchen, L. Davi, A.-R. Sadeghi, and T. Holz. 2015. Counterfeit object-oriented programming: On the difficulty of preventing code reuse attacks in C++ applications. In *36th IEEE Symposium on Security and Privacy (S&P)*, pp. 745–762. DOI: 10.1109/SP.2015.51. 15, 20, 67, 70, 97, 140, 182, 183, 184, 185, 186, 200, 204, 211
- F. Schuster, T. Tendyck, J. Powny, A. Maaß, M. Stegmanns, M. Contag, and T. Holz. 2014. Evaluating the effectiveness of current anti-ROP defenses. In *Research in Attacks, Intrusions, and Defenses*, volume 8688 of *Lecture Notes in Computer Science*. DOI: 10.1007/978-3-319-11379-1_5. 139, 140, 177, 182, 186, 188, 209
- E. J. Schwartz, T. Avgerinos, and D. Brumley. 2010. All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask). In *IEEE Symposium on Security and Privacy (S&P)*. DOI: 10.1109/SP.2010.26. 86
- E. J. Schwartz, T. Avgerinos, and D. Brumley. 2011. Q: Exploit hardening made easy. In *Proceedings of the 20th USENIX Conference on Security (SEC)*, pp. 25–25. 136
- C. Segulja and T. S. Abdelrahman. 2014. What is the cost of weak determinism? In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, pp. 99–112. DOI: 10.1145/2628071.2628099. 254
- J. Seibert, H. Okhravi, and E. Söderström. 2014. Information leaks without memory disclosures: Remote side channel attacks on diversified code. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pp. 54–65. DOI: 10.1145/2660267.2660309. 141, 182
- K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov. 2012. AddressSanitizer: A fast address sanity checker. In *USENIX Annual Technical Conference*, pp. 309–318. 82, 84, 173, 178
- F. J. Serna. 2012. CVE-2012-0769, the case of the perfect info leak. http://media.blackhat.com/bh-us-12/Briefings/Serna/BH_US_12_Serna_Leak_Era_Slides.pdf. 63, 117
- H. Shacham. 2007. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, pp. 552–561. DOI: 10.1145/1315245.1315313. 62, 172, 184, 185, 186, 200, 233

- H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh. 2004. On the effectiveness of address-space randomization. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pp. 298–307. DOI: [10.1145/1030083.1030124](https://doi.org/10.1145/1030083.1030124). 62
- N. Shavit and D. Touitou. 1995. Software transactional memory. In *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 204–213. 28
- K. Z. Snow, F. Monrose, L. Davi, A. Dmitrienko, C. Liebchen, and A.-R. Sadeghi. 2013. Just-in-time code reuse: On the effectiveness of fine-grained address space layout randomization. In *34th IEEE Symposium on Security and Privacy (S&P)*, pp. 574–588. DOI: [10.1109/SP.2013.45](https://doi.org/10.1109/SP.2013.45). 10, 20, 49, 63, 82, 86, 117, 141, 177, 182, 184, 186, 203
- K. Z. Snow, R. Rogowski, J. Werner, H. Koo, F. Monrose, and M. Polychronakis. 2016. Return to the zombie gadgets: Undermining destructive code reads via code inference attacks. In *37th IEEE Symposium on Security and Privacy (S&P)*, pp. 954–968. DOI: [10.1109/SP.2016.61](https://doi.org/10.1109/SP.2016.61). 70
- Solar Designer. 1997a. “return-to-libc” attack. Bugtraq. 203
- Solar Designer. 1997b. lpr LIBC RETURN exploit. <http://insecure.org/spl0its/linux.libc.return.lpr.spl0it.html>. 203
- C. Song, C. Zhang, T. Wang, W. Lee, and D. Melski. 2015. Exploiting and protecting dynamic code generation. In *Network and Distributed System Security Symposium (NDSS)*. 49, 58, 60
- A. Sotirov. 2007. Heap feng shui in JavaScript. In *Proceedings of Black Hat Europe*. 132
- E. H. Spafford. January 1989. The internet worm program: An analysis. *SIGCOMM Comput. Commun. Rev.*, 19 (1): 17–57. ISSN 0146-4833. DOI: [10.1145/66093.66095](https://doi.org/10.1145/66093.66095). 61
- SPARC. SPARC V8 processor. <http://www.sparc.org>. 206
- R. Strackx, Y. Younan, P. Philippaerts, F. Piessens, S. Lachmund, and T. Walter. 2009. Breaking the memory secrecy assumption. In *2nd European Workshop on System Security (EUROSEC)*, pp. 1–8. DOI: [10.1145/1519144.1519145](https://doi.org/10.1145/1519144.1519145). 63, 117
- D. Sullivan, O. Arias, L. Davi, P. Larsen, A.-R. Sadeghi, and Y. Jin. 2016. Strategy without tactics: Policy-agnostic hardware-enhanced control-flow integrity. In *IEEE/ACM Design Automation Conference (DAC)*, pp. 83.2:1–6. DOI: [10.1145/2897937.2898098](https://doi.org/10.1145/2897937.2898098). 209
- L. Szekeres, M. Payer, T. Wei, and D. Song. 2013. SoK: Eternal war in memory. In *Proceedings International Symposium on Security and Privacy (S&P)*. DOI: [10.1109/SP.2013.13](https://doi.org/10.1109/SP.2013.13). 2, 61, 82, 85, 211
- L. Szekeres, M. Payer, L. Wei, D. Song, and R. Sekar. 2014. Eternal war in memory. *IEEE Security and Privacy Magazine*. DOI: [10.1109/MSP.2013.47](https://doi.org/10.1109/MSP.2013.47). 2
- A. Tang, S. Sethumadhavan, and S. Stolfo. 2015. Heisenbyte: Thwarting memory disclosure attacks using destructive code reads. In *ACM Conference on Computer and Communications Security (CCS)*, pp. 256–267. DOI: [10.1145/2810103.2813685](https://doi.org/10.1145/2810103.2813685). 70

- C. Tice, T. Roeder, P. Collingbourne, S. Checkoway, Ú. Erlingsson, L. Lozano, and G. Pike. 2014. Enforcing forward-edge control-flow integrity in GCC & LLVM. In *Proceedings of the 23rd USENIX Security Symposium*. <http://dl.acm.org/citation.cfm?id=2671225.2671285>. 58, 86, 173, 175, 182, 204, 208, 211, 233
- M. Tran, M. Etheridge, T. Bletsch, X. Jiang, V. Freeh, and P. Ning. 2011. On the expressiveness of return-into-libc attacks. In *Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection (RAID)*, pp. 121–141. DOI: [10.1007/978-3-642-23644-0_7](https://doi.org/10.1007/978-3-642-23644-0_7). 117, 140, 183, 184, 185, 200, 204
- A. van de Ven. August 2004. New security enhancements in Red Hat Enterprise Linux v.3, update 3. http://people.redhat.com/mingo/exec-shield/docs/WHP0006US_Execshield.pdf. 9, 82
- V. van der Veen, D. Andriesse, E. Göktaş, B. Gras, L. Sambuc, A. Slowinska, H. Bos, and C. Giuffrida. 2015. PathArmor: Practical ROP protection using context-sensitive CFI. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 927–940. DOI: [10.1145/2810103.2813673](https://doi.org/10.1145/2810103.2813673). 14, 137
- V. van der Veen, N. D. Sharma, L. Cavallaro, and H. Bos. 2012. Memory errors: The past, the present, and the future. In *Proceedings of the 15th International Conference on Research in Attacks, Intrusions, and Defenses (RAID)*, pp. 86–106. DOI: [10.1007/978-3-642-33338-5_5](https://doi.org/10.1007/978-3-642-33338-5_5). 61
- S. Volckaert. 2015. Advanced Techniques for multi-variant execution. Ph.D. thesis, Ghent University. 226, 231
- S. Volckaert, B. Coppens, and B. De Sutter. 2015. Cloning your gadgets: Complete ROP attack immunity with multi-variant execution. *IEEE Trans. on Dependable and Secure Computing*, 13 (4): 437–450. DOI: [10.1109/TDSC.2015.2411254](https://doi.org/10.1109/TDSC.2015.2411254). 211, 250
- S. Volckaert, B. Coppens, B. De Sutter, K. De Bosschere, P. Larsen, and M. Franz. 2017. Taming parallelism in a multi-variant execution environment. In *Proceedings of the 12th European Conference on Computer Systems (EuroSys)*, pp. 270–285. DOI: [10.1145/3064176.3064178](https://doi.org/10.1145/3064176.3064178). 230, 232
- S. Volckaert, B. Coppens, A. Voulimeneas, A. Homescu, P. Larsen, B. De Sutter, and M. Franz. 2016. Secure and efficient application monitoring and replication. In *USENIX Annual Technical Conference (ATC)*, pp. 167–179. 214, 215, 247
- S. Volckaert, B. De Sutter, T. De Baets, and K. De Bosschere. 2013. GHUMVEE: Efficient, effective, and flexible replication. In *5th International Symposium on Foundations and Practice of Security (FPS)*, pp. 261–277. 214, 217, 232
- R. Wahbe, S. Lucco, T. Anderson, and S. Graham. 1993. Efficient software-based fault isolation. In *Proceedings of the 14th ACM Symposium on Operating System Principles*, pp. 203–216. DOI: [10.1145/168619.168635](https://doi.org/10.1145/168619.168635). 8, 9, 41, 68, 249
- Z. Wang and X. Jiang. 2010. HyperSafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pp. 380–395. DOI: [10.1109/SP.2010.30](https://doi.org/10.1109/SP.2010.30). 58, 208

- R. Wartell, V. Mohan, K. W. Hamlen, and Z. Lin. 2012. Binary stirring: Self-randomizing instruction addresses of legacy x86 binary code. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 157–168. DOI: [10.1145/2382196.2382216](https://doi.org/10.1145/2382196.2382216). 11, 173, 177
- R. N. M. Watson, J. Anderson, B. Laurie, and K. Kennaway. 2010. Capsicum: Practical capabilities for UNIX. In *19th USENIX Security Symposium*, pp. 29–46. 16
- T. Wei, T. Wang, L. Duan, and J. Luo. 2011. INSeRT: Protect dynamic code generation against spraying. In *International Conference on Information Science and Technology (ICIST)*, pp. 323–328. DOI: [10.1109/ICIST.2011.5765261](https://doi.org/10.1109/ICIST.2011.5765261). 59
- J. Werner, G. Baltas, R. Dallara, N. Otternes, K. Snow, F. Monrose, and M. Polychronakis. 2016. No-execute-after-read: Preventing code disclosure in commodity software. In *11th ACM Symposium on Information, Computer, and Communications Security (ASIACCS)*, pp. 35–46. DOI: [10.1145/2897845.2897891](https://doi.org/10.1145/2897845.2897891). 70
- J. Wilander, N. Nikiforakis, Y. Younan, M. Kamkar, and W. Joosen. 2011. RIPE: Runtime intrusion prevention evaluator. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 41–50. DOI: [10.1145/2076732.2076739](https://doi.org/10.1145/2076732.2076739). 109, 239
- R. Wojtczuk. 1998. Defeating Solar Designer's non-executable stack patch. <http://insecure.org/sploits/non-executable.stack.problems.html>. 20, 81, 82, 203
- C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman. 2002. Linux security modules: General security support for the Linux kernel. In *Proceedings 11th USENIX Security Symposium*. 16
- R. Wu, P. Chen, B. Mao, and L. Xie. 2012. RIM: A method to defend from JIT spraying attack. In *7th International Conference on Availability, Reliability, and Security (ARES)*, pp. 143–148. DOI: [10.1109/ARES.2012.11](https://doi.org/10.1109/ARES.2012.11). 59
- Y. Xia, Y. Liu, H. Chen, and B. Zang. 2012. CFIMon: Detecting violation of control flow integrity using performance counters. In *IEEE/IFIP Conference on Dependable Systems and Networks (DSN)*, pp. 1–12. DOI: [10.1109/DSN.2012.6263958](https://doi.org/10.1109/DSN.2012.6263958). 173, 176
- F. Yao, J. Chen, and G. Venkataramani. 2013. JOP-alarm: Detecting jump-oriented programming-based anomalies in applications. In *IEEE 31st International Conference on Computer Design (ICCD)*, pp. 467–470. DOI: [10.1109/ICCD.2013.6657084](https://doi.org/10.1109/ICCD.2013.6657084). 209
- B. Yee, D. Sehr, G. Dardyk, J. B. Chen, R. Muth, T. Ormandy, S. Okasaka, N. Narula, and N. Fullagar. 2009. Native client: A sandbox for portable, untrusted x86 native code. In *30th IEEE Symposium on Security and Privacy (S&P)*, pp. 79–93. DOI: [10.1109/SP.2009.25](https://doi.org/10.1109/SP.2009.25). 8, 39, 86
- B. Zeng, G. Tan, and Ú. Erlingsson. 2013. Strato: A retargetable framework for low-level inlined-reference monitors. In *USENIX Security Symposium*, pp. 369–382. 58, 86
- B. Zeng, G. Tan, and G. Morrisett. 2011. Combining control-flow integrity and static analysis for efficient and validated data sandboxing. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 29–40. DOI: [10.1145/2046707.2046713](https://doi.org/10.1145/2046707.2046713). 58, 59, 86

- C. Zhang, C. Song, K. Z. Chen, Z. Chen, and D. Song. 2015. VTint: Defending virtual function tables' integrity. In *Symposium on Network and Distributed System Security (NDSS)*. DOI: [10.14722/ndss.2015.23099](https://doi.org/10.14722/ndss.2015.23099) . 160, 173, 176, 182
- C. Zhang, T. Wei, Z. Chen, L. Duan, L. Szekeres, S. McCamant, D. Song, and W. Zou. 2013. Practical control flow integrity and randomization for binary executables. In *34th IEEE Symposium on Security and Privacy (S&P)*, pp. 559–573. DOI: [10.1109/SP.2013.44](https://doi.org/10.1109/SP.2013.44). 38, 82, 86, 97, 110, 114, 117, 118, 119, 127, 136, 169, 173, 174, 182, 208, 209
- M. Zhang and R. Sekar. 2013. Control flow integrity for COTS binaries. In *Proceedings of the 22nd USENIX Security Symposium*, pp. 337–352. <http://dl.acm.org/citation.cfm?id=2534766.2534796>. 38, 82, 86, 97, 110, 114, 117, 118, 119, 127, 136, 173, 174, 182, 208, 209
- H. W. Zhou, X. Wu, W. C. Shi, J. H. Yuan, and B. Liang. 2014. HDROP: Detecting ROP attacks using performance monitoring counters. In *Information Security Practice and Experience*, pp. 172–186. Springer International Publishing. DOI: [10.1007/978-3-319-06320-1_14](https://doi.org/10.1007/978-3-319-06320-1_14). 173, 177
- X. Zhou, K. Lu, X. Wang, and . Li. 2012. Exploiting parallelism in deterministic shared memory multiprocessing. *Journal of Parallel and Distributed Computing*, 72(5):716–727. DOI: [10.1016/j.jpdc.2012.02.008](https://doi.org/10.1016/j.jpdc.2012.02.008). 230

Contributor Biographies

Editors

Per Larsen is trying his hand as an entrepreneur and co-founded an information security startup—Immunant, Inc.—specializing in exploit mitigation. Previously, he worked for four years as a postdoctoral scholar at the University of California, Irvine. He graduated with a Ph.D. from the Technical University of Denmark in 2011.

Per co-organized the 2015 Dagstuhl Seminar upon which this book is based and has served as program committee member for several academic conferences including USENIX Security, USENIX WOOT, ICDCS, and AsiaCCS. In 2015, he was recognized as a DARPA Riser.

Ahmad Sadeghi is a full professor of Computer Science at TU Darmstadt, Germany, and the Director of the Intel Collaborative Research Institute for Secure Computing (ICRI-SC) at TU Darmstadt. He holds a Ph.D. in computer science from the University of Saarland in Saarbrücken, Germany. Prior to academia, he worked in research and development for telecommunications enterprises, amongst others Ericsson Telecommunications. He is editor-in-chief of *IEEE Security and Privacy Magazine*, and on the editorial board of ACM Books. He served five years on the editorial board of the *ACM Transactions on Information and System Security* (TISSEC), and was guest editor of the *IEEE Transactions on Computer-Aided Design (Special Issue on Hardware Security and Trust)*.

Authors

Orlando Arias is pursuing his Ph.D. in computer engineering from the University of Central Florida under the advisement of Dr. Yier Jin. His research interests include secure computer architectures, network security, IP core design, and integration and cryptosystems.

Elias Athanasopoulos is an assistant professor with the Computer Science Department at the University of Cyprus. Before joining the University of Cyprus, he was an assistant professor with Vrije Universiteit Amsterdam. He holds a BSc in physics

from the University of Athens and a Ph.D. in computer science from the University of Crete. Elias is a Microsoft Research Ph.D. Scholar. He has interned with Microsoft Research in Cambridge and worked as a research assistant with FORTH in Greece from 2005 to 2011. Elias is also a Marie Curie fellow. Before joining the faculty of Vrije Universiteit Amsterdam, he was a postdoctoral research scientist with Columbia University and a collaborating researcher with FORTH.

Herbert Bos is a professor of systems and network security at Vrije Universiteit Amsterdam, where he heads the VUsec research group. He obtained his Ph.D. from Cambridge University Computer Laboratory (UK). Coming from a systems background, he drifted into security a few years ago and never left.

George Candea heads the Dependable Systems Lab at EPFL, where he conducts research on both the fundamentals and the practice of achieving reliability and security in complex software systems. His main focus is on real-world large-scale systems—millions of lines of code written by hundreds of programmers—because going from a small program to a large system introduces fundamental challenges that cannot be addressed with the techniques that work at small scale. George is also Chairman of Cyberhaven, a cybersecurity company he co-founded with his former students to defend sensitive data against advanced attacks, social engineering, and malicious insiders. In the past, George was CTO and later Chief Scientist of Aster Data Systems (now Teradata Aster). Before that, he held positions at Oracle, Microsoft Research, and IBM Research. George is a recipient of the first Eurosys Jochen Liedtke Young Researcher Award (2014), an ERC StG award (2011), and the MIT TR35 Young Innovators award (2005). He received his Ph.D. (2005) in computer science from Stanford and his B.S. (1997) and M.Eng. (1998) in electrical engineering and computer science from MIT.

Bart Coppens is a postdoctoral researcher at Ghent University in the Computer Systems Lab. He received his Ph.D. in computer science engineering from the Faculty of Engineering and Architecture at Ghent University in 2013. His research focuses on protecting software against different forms of attacks using compiler-based techniques and run-time techniques.

Stephen Crane started seriously diving into security during his undergrad at Cal Poly Pomona, competing in CCDC. From there he worked on research somewhere in the intersection of systems security and compilers at UC Irvine. After transforming into Dr. Crane, Stephen founded Immunant with fellow UCI researchers, where he tries to get exploit mitigation tools into the hands of developers.

Lucas Davi is an assistant professor of computer science at University of Duisburg-Essen, Germany, and associated researcher at the Intel Collaborative Research

Institute for Secure Computing (ICRI-SC) at TU Darmstadt, Germany. He received his Ph.D. in computer science from TU Darmstadt. His research focus includes system security, software security, and trusted computing. His Ph.D. thesis on code-reuse attacks and defenses has been awarded with the ACM SIGSAC Dissertation Award 2016.

Bjorn De Sutter is a professor at Ghent University in the Computer Systems Lab. He obtained his MSc. and Ph.D. degrees in computer science from Ghent University's Faculty of Engineering in 1997 and 2002. His research focuses on the use of compiler techniques and run-time techniques to aid programmers with non-functional aspects of their software, such as performance, code size, reliability, and software protection. He has published over 80 peer-reviewed papers on these topics.

Michael Franz is the director of the Secure Systems and Software Laboratory at the University of California, Irvine (UCI). He is a full professor of computer science in UCI's Donald Bren School of Information and Computer Sciences and a full professor of electrical engineering and computer science (by courtesy) in UCI's Henry Samueli School of Engineering. Prof. Franz was an early pioneer in the areas of mobile code and dynamic compilation. He created an early just-in-time compilation system, contributed to the theory and practice of continuous compilation and optimization, and co-invented the trace compilation technology that eventually became the JavaScript engine in Mozilla's Firefox browser. Franz received a Dr. sc. techn. degree in computer science and a Dipl. Informatik-Ing. ETH degree, both from the Swiss Federal Institute of Technology, ETH Zurich.

Enes Göktaş is a Ph.D. Student in the systems and network security group at the Vrije Universiteit Amsterdam. His research focus is on evaluating and developing mitigations against memory corruption vulnerabilities. His previous work includes evaluation and proposal of Control-Flow Integrity based mitigations. His interests lie in the area of software security, as well as binary analysis and instrumentation.

Thorsten Holz is a professor in the Faculty of Electrical Engineering and Information Technology at Ruhr-University Bochum, Germany. His research interests include systems-oriented aspects of secure systems, with a specific focus on applied computer security. Currently, his work concentrates on bots/botnets, automated analysis of malicious software, and studying the latest attack vectors. He received the Dipl.-Inform. degree in computer science from RWTH Aachen, Germany (2005), and a Ph.D. degree from University of Mannheim (2009). Prior to joining Ruhr-University Bochum in April 2010, he was a postdoctoral researcher in the Automation Systems Group at the Technical University of Vienna, Austria. In

2011, Thorsten received the Heinz Maier-Leibnitz Prize from the German Research Foundation (DFG).

Andrei Homescu finished his doctoral studies at UC Irvine in 2015, after which he co-founded Immunant with three of the present co-authors. Andrei has published widely in the areas of systems security, language runtimes, and exploit mitigation. He also led the development of selfrando, a production-ready, open-source randomization engine, and is the lead author on several US patents and patent applications.

Yier Jin received his Ph.D. in electrical engineering from Yale University in 2012. He is an assistant professor in the Department of Electrical and Computer Engineering, University of Central Florida, USA. His research interests include hardware security, IoT security, and formal methods. He is a member of IEEE and ACM.

Volodymyr Kuznetsov is a security researcher who focuses on practical applications of program analysis and other formal methods in systems security, with particular interest in protecting sensitive data in applications and systems with security vulnerabilities. Volodymyr's research was released as open source projects that have become widely used in research community and industry (<http://s2e.epfl.ch/>, <http://clang.llvm.org/docs/SafeStack.html>, and others) and were recognized with an open source award. Volodymyr obtained his Ph.D. in Computer Science at EPFL in 2016 and now leads a cyber security company that brings state-of-the-art research into the world of enterprise cyber security.

Ben Niu earned his Ph.D. degree in computer science from Lehigh University, USA, in 2016, advised by professor Gang Tan. He is currently a security software engineer at Microsoft Corporation. His research interests are system security and parallel programming.

Hamed Okhravi is a senior staff member at the Cyber Analytics and Decision Systems Group of MIT Lincoln Laboratory, where he leads programs and conducts research in the area of systems security. His research interests include cyber security, science of security, security evaluation, and operating systems. He is the recipient of the 2014 MIT Lincoln Laboratory Early Career Technical Achievement Award and 2015 Team Award for his work on cyber moving target research. He is also the recipient of an honorable mention (runner-up) at the 2015 NSA's 3rd Annual Best Scientific Cybersecurity Paper Competition. Currently, his research is focused on analyzing and developing system security defenses.

He has served as a program chair for the ACM CCS Moving Target Defense (MTD) workshop and program committee member for a number of academic conferences and workshops including ACM CCS, NDSS, RAID, AsiaCCS, ACNS, and IEEE SecDev.

Dr. Okhravi earned his MS and Ph.D. in electrical and computer engineering from University of Illinois at Urbana-Champaign in 2006 and 2010, respectively.

Mathias Payer is a security researcher and assistant professor in computer science at Purdue University, leading the HexHive group. His research focuses on protecting applications even in the presence of vulnerabilities, with a focus on memory corruption. He is interested in system security, binary exploitation, user-space software-based fault isolation, binary translation/recompilation, and (application) virtualization. All implementation prototypes from his group are open source. In 2014, he founded the b01lers Purdue CTF team. Before joining Purdue in 2014, he spent two years as a postdoc in Dawn Song's BitBlaze group at UC Berkeley. He graduated from ETH Zurich with a Dr. sc. ETH in 2012.

Georgios Portokalidis is an assistant professor in the Department of Computer Science at Stevens Institute of Technology. He obtained his Ph.D. from Vrije Universiteit in Amsterdam in February 2010, and also spent a couple of years as a postdoc at Columbia University in New York. His research interests center mainly around the area of systems and security, including software and network security, authentication, privacy, and software resiliency. His recent work has revolved around code-reuse attacks, efficient information-flow tracking, and security applications using the Internet of Things. During his Ph.D. he worked on the Argos emulator, a platform for hosting high-interaction honeypots that can automatically detect zero-day control-flow hijacking attacks, and Paranoid Android, a record-replay system for the Android OS.

Felix Schuster has been a researcher at the Microsoft Research Cambridge (UK) lab since 2015. Before joining Microsoft Research, he obtained a Ph.D. from Ruhr-Universität Bochum. Felix is broadly interested in applied systems and software security and is part of the lab's Constructive Security Group. In the past, he worked on topics like code-reuse attacks and defenses (e.g., COOP) and automated binary code analysis. Currently, Felix's research focusses on the design of practical solutions for the trusted cloud like the VC3 system or the Coco blockchain framework.

R. Sekar is a Professor of Computer Science and the Director of the Secure Systems Laboratory and the Center for Cyber Security at Stony Brook University. He received his Bachelor's degree in Electrical Engineering from IIT, Madras (India) in 1986, and his Ph.D. in Computer Science from Stony Brook in 1991. He then served as a Research Scientist at Bellcore until 1996, and then as faculty at Iowa State University. Sekar's research interests are focused on software exploit detection and mitigation, malware and untrusted code defense, and security policies and their enforcement.

Dawn Song is a professor in the Department of Electrical Engineering and Computer Science at UC Berkeley. Her research interest lies in deep learning and security. She has studied diverse security and privacy issues in computer systems and networks, including areas ranging from software security, networking security, database security, distributed systems security, and applied cryptography, to the intersection of machine learning and security. She is the recipient of various awards including the MacArthur Fellowship, the Guggenheim Fellowship, the NSF CAREER Award, the Alfred P. Sloan Research Fellowship, the MIT Technology Review TR-35 Award, the George Tallman Ladd Research Award, the Okawa Foundation Research Award, the Li Ka Shing Foundation Women in Science Distinguished Lecture Series Award, the Faculty Research Award from IBM, Google and other major tech companies, and Best Paper Awards from top conferences. She obtained her Ph.D. from UC Berkeley. Prior to joining UC Berkeley as a faculty, she was an Assistant Professor at Carnegie Mellon University from 2002–2007.

Dean Sullivan is pursuing his Ph.D. in computer engineering from the University of Central Florida under the advisement of Dr. Yier Jin. His research interests include system security and computer architecture.

László Szekeres is a software security researcher at Google. He works on developing techniques for protecting against security bugs, primarily in C/C++ code. His research is focused on finding and hardening against vulnerabilities using automated test generation, program analysis, compiler techniques, and machine learning. He obtained his Ph.D. in Computer Science from Stony Brook University in 2017. During his studies he spent a year as a visiting researcher at UC Berkeley. In 2010, he was awarded the Fulbright Foreign Student Scholarship. Before returning to academia for his doctorate degree, he led a security research team at a spin-off company of the Budapest University of Technology and Economics.

Gang Tan received his Ph.D. in computer science from Princeton University in 2005. He is an associate professor in the Department of Computer Science and Engineering, Pennsylvania State University, USA. His research interests include software security, programming languages, and formal methods. He is a member of IEEE and ACM.

Stijn Volckaert received his Ph.D. degree from Ghent University's Faculty of Engineering and Architecture. He is currently a postdoctoral scholar in the Department of Computer Science at the University of California, Irvine. His research interests include security, operating systems, and software protection.